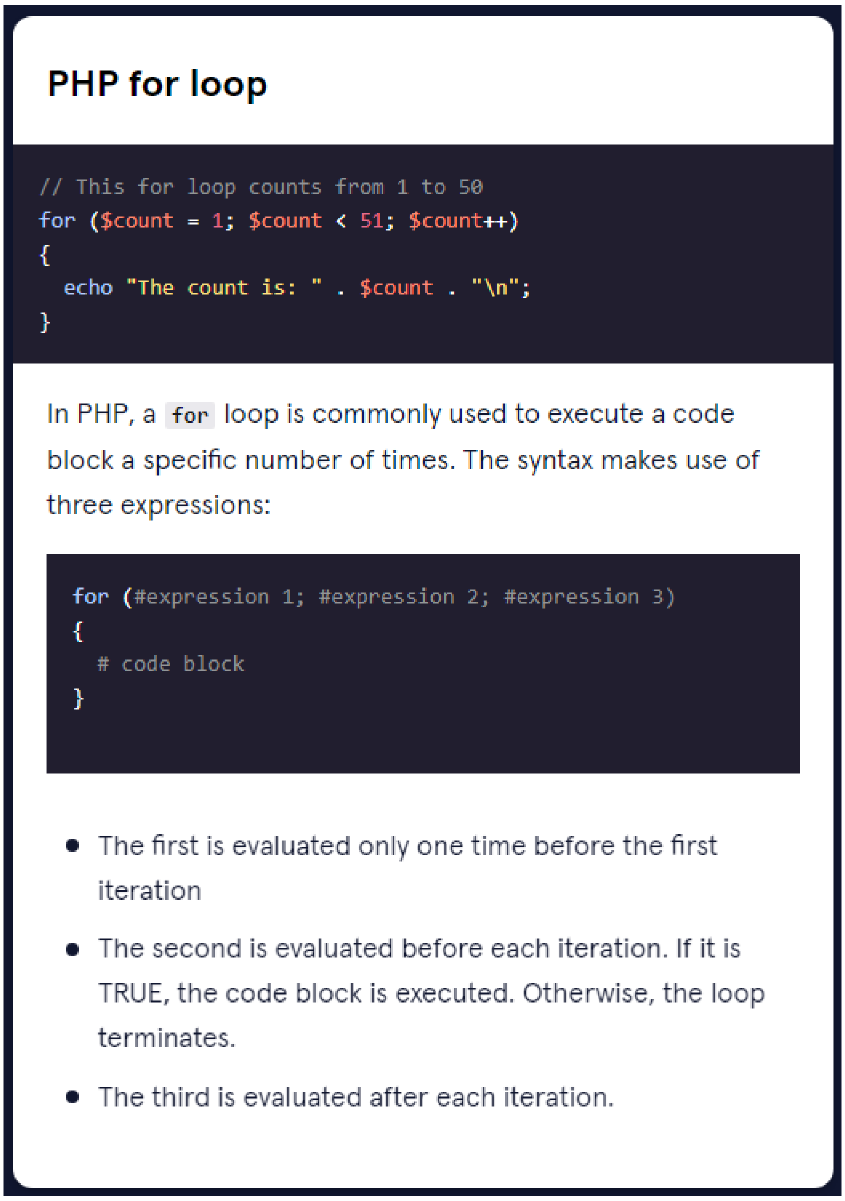
MODULE PRACTICE



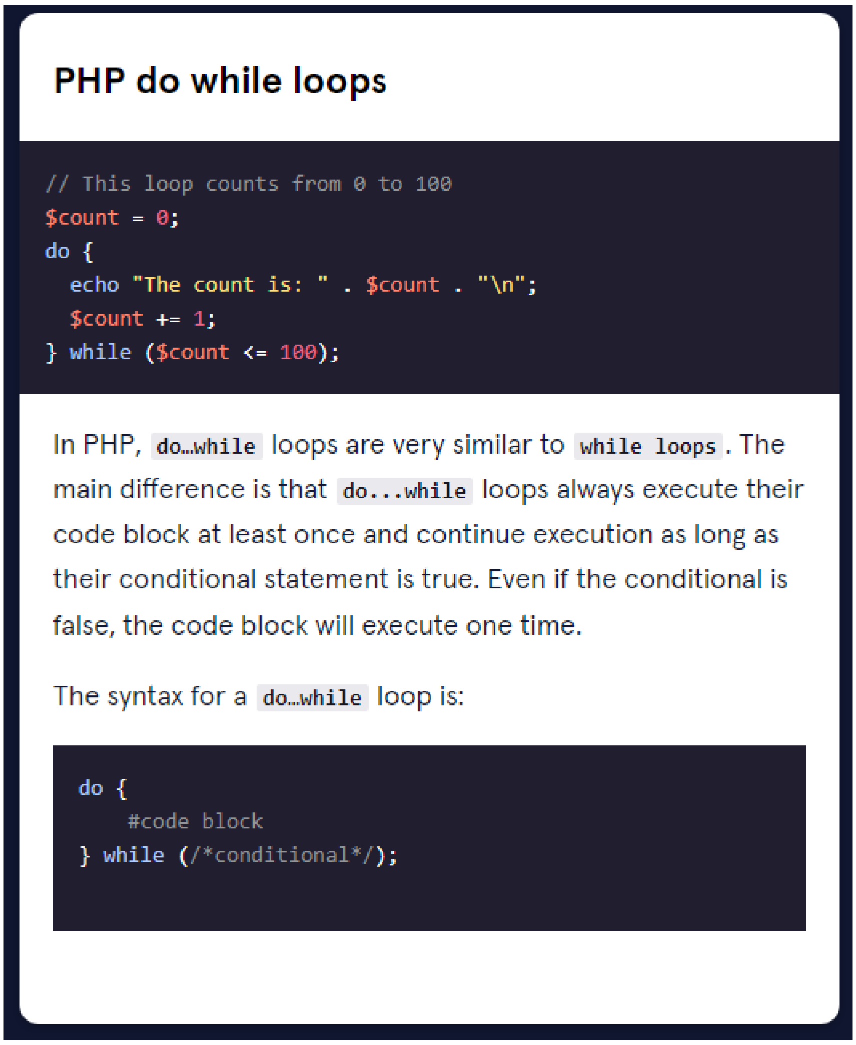
PHP for loop

```
// This for loop counts from 1 to 50
for ($count = 1; $count < 51; $count++)
{
  echo "The count is: " . $count . "\n";
}
```

In PHP, a `for` loop is commonly used to execute a code block a specific number of times. The syntax makes use of three expressions:

```
for (#expression 1; #expression 2; #expression 3)
{
  # code block
}
```

- The first is evaluated only one time before the first iteration
- The second is evaluated before each iteration. If it is TRUE, the code block is executed. Otherwise, the loop terminates.
- The third is evaluated after each iteration.



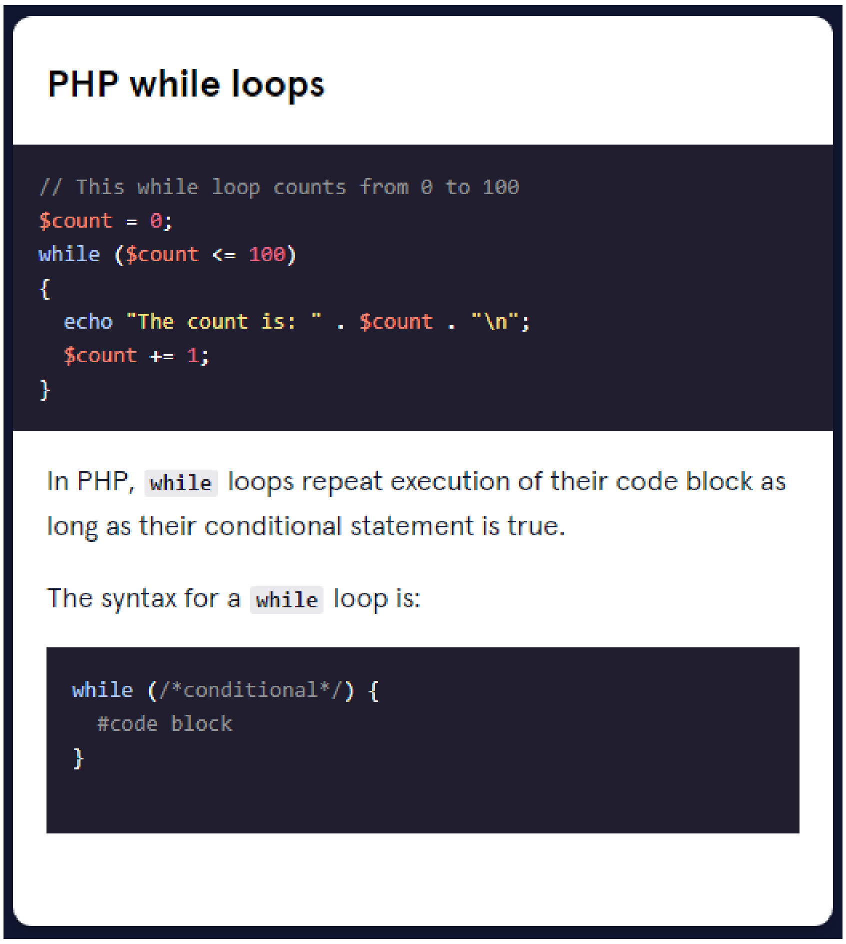
PHP do while loops

```
// This loop counts from 0 to 100
$count = 0;
do {
  echo "The count is: " . $count . "\n";
  $count += 1;
} while ($count <= 100);
```

In PHP, `do…while` loops are very similar to `while loops`. The main difference is that `do...while` loops always execute their code block at least once and continue execution as long as their conditional statement is true. Even if the conditional is false, the code block will execute one time.

The syntax for a `do…while` loop is:

```
do {
    #code block
} while (/*conditional*/);
```



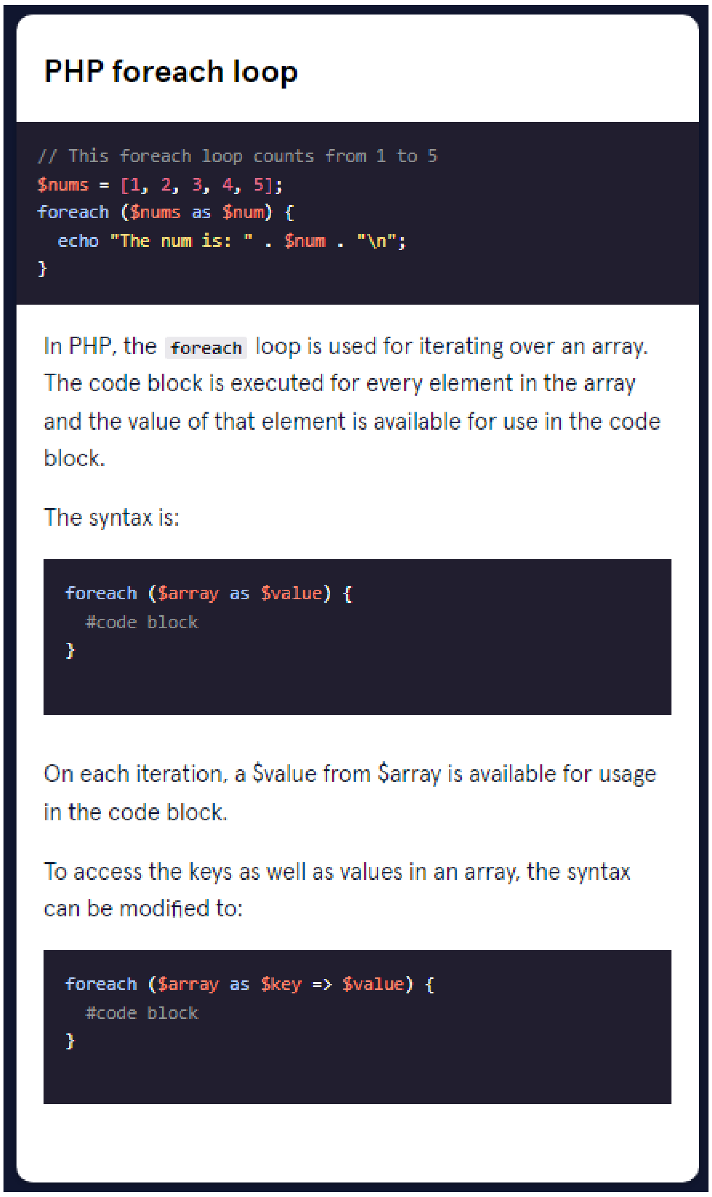
PHP while loops

```php
// This while loop counts from 0 to 100
$count = 0;
while ($count <= 100)
{
  echo "The count is: " . $count . "\n";
  $count += 1;
}
```

In PHP, `while` loops repeat execution of their code block as long as their conditional statement is true.

The syntax for a `while` loop is:

```php
while (/*conditional*/) {
  #code block
}
```



PHP foreach loop

```php
// This foreach loop counts from 1 to 5
$nums = [1, 2, 3, 4, 5];
foreach ($nums as $num) {
  echo "The num is: " . $num . "\n";
}
```

In PHP, the `foreach` loop is used for iterating over an array. The code block is executed for every element in the array and the value of that element is available for use in the code block.

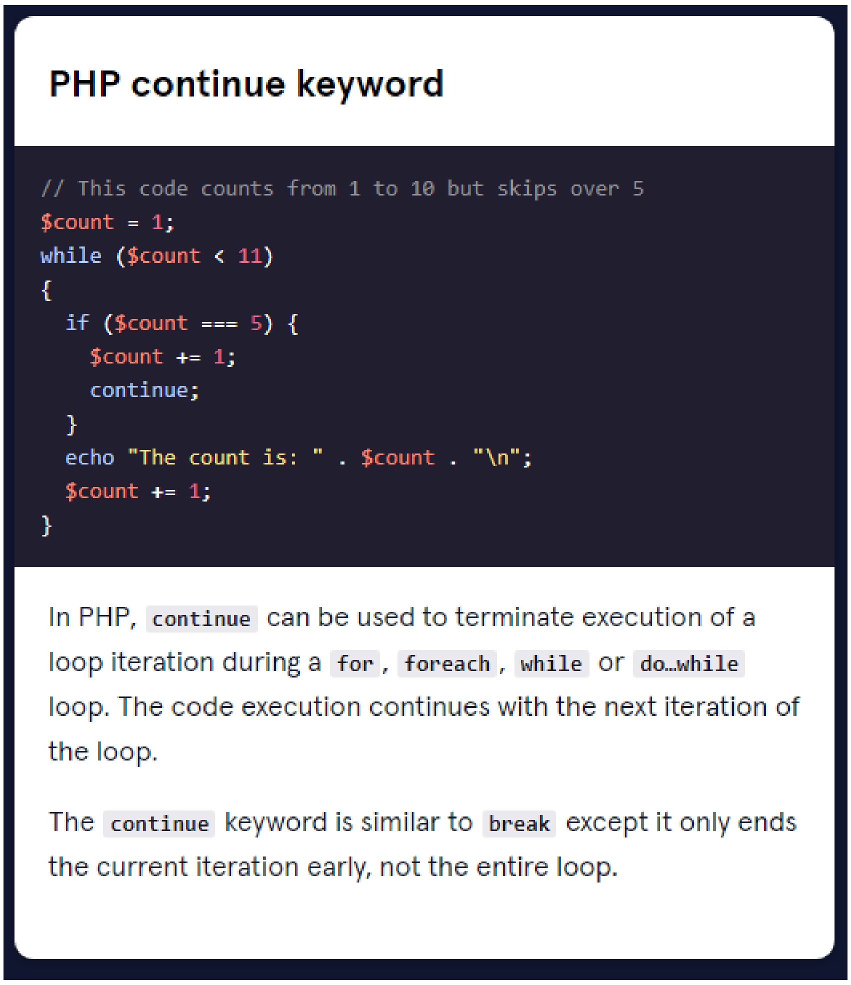
The syntax is:

```php
foreach ($array as $value) {
  #code block
}
```

On each iteration, a $value from $array is available for usage in the code block.

To access the keys as well as values in an array, the syntax can be modified to:

```php
foreach ($array as $key => $value) {
  #code block
}
```



PHP continue keyword

```php
// This code counts from 1 to 10 but skips over 5
$count = 1;
while ($count < 11)
{
  if ($count === 5) {
    $count += 1;
    continue;
  }
  echo "The count is: " . $count . "\n";
  $count += 1;
}
```

In PHP, `continue` can be used to terminate execution of a loop iteration during a `for`, `foreach`, `while` or `do…while` loop. The code execution continues with the next iteration of the loop.

The `continue` keyword is similar to `break` except it only ends the current iteration early, not the entire loop.

PHP break keyword

```php
// We can use the break statement to end the loop once the
count reaches 4
$count = 1;
while ($count < 10)
{
  echo "The count is: " . $count . "\n";
  if ($count === 4) {
    break;
  }
  $count += 1;
}
```

In PHP, `break` can be used to terminate execution of a `for`, `foreach`, `while` or `do…while` loop.

One downside of heavy usage of break statements is that code can become less readable.